	Top	Module1	Module2
		U0	U1
SigName1	I	I	
Bundle1 — sigA		O	in1
Bundle1 — sigB		O	
SigName2	O		z

Figure 1
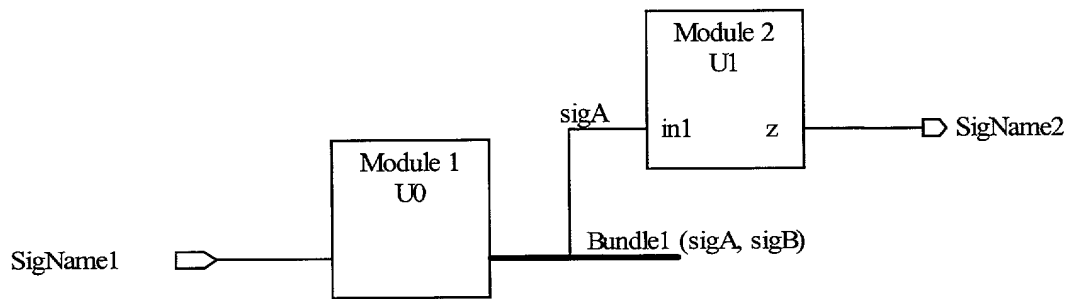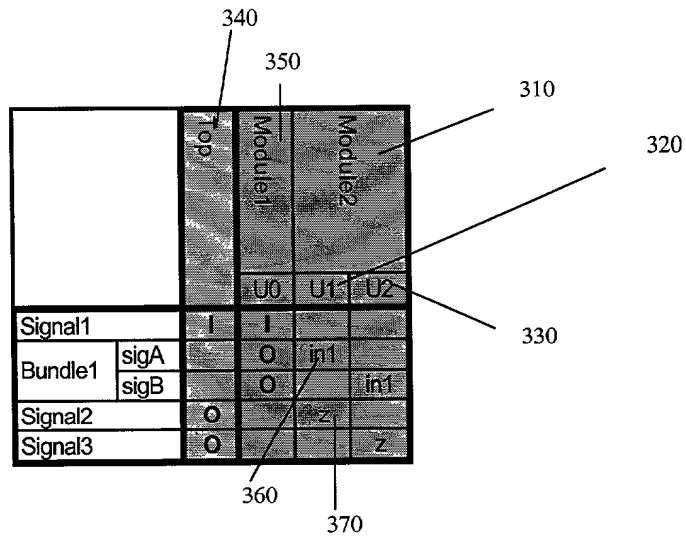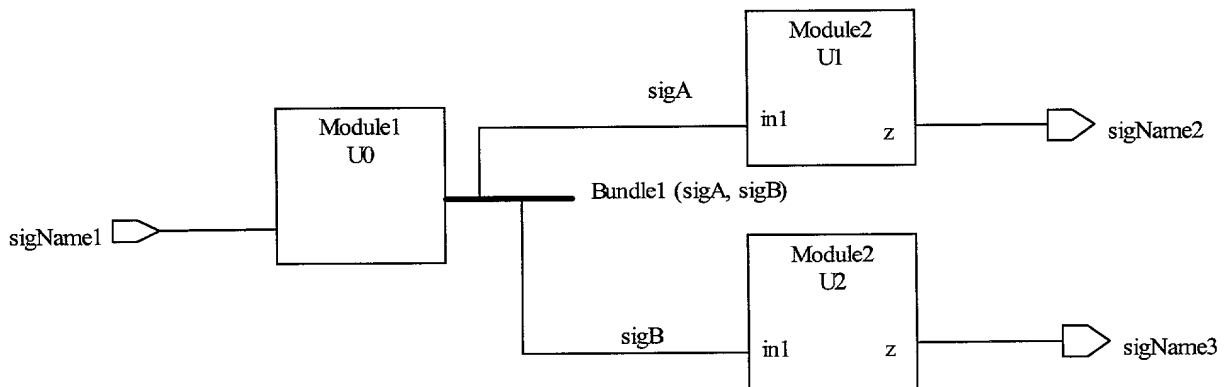

Figure 2

Figure 3

Figure 4

```vhdl
503   LIBRARY ieee;
504   USE ieee.std_logic.all;
505
506   ENTITY Top IS
507           port    (Signal1 : IN std_logic;
508                   Signal2 : OUT std_logic
509                   Signal3 : OUT std_logic);
510   END Top;
511
512   ARCHITECTURE struct OF Top IS
513
514           SIGNAL sigA              : std_logic;
515           SIGNAL sigB              : std_logic;
516
517           COMPONENT Module1
518           PORT(
519               Signal1       : IN std_logic;
520               sigA          :OUT std_logic;
521               sigB          :OUT std_logic;
522           );
523           END COMPONENT;
524
525           COMPONENT Module2
526           PORT(
527               in1     : IN    std_logic;
528               z       : OUT std_logic;
529           );
530           END COMPONENT;
531
532   BEGIN
533
534           U0 : Module1
535               PORT MAP(
536                   sigA        => sigA,
537                   sigB        => sigB,
538                   Signal1     => Signal1
539               );
540
541           U1 : Module2
542               PORT MAP(
543                   in1 => sigA,
544                   z   => Signal2
545               );
546
```

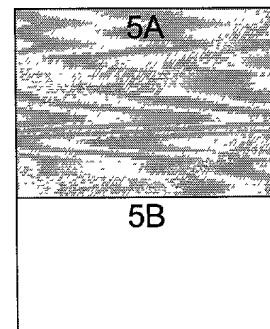Figure 5

Figure 5A

```
546        U2 : Module2
547           PORT MAP(
548             in1 => sigB,
549             z   => Signal3
550           );
551
552
553     END ARCHITECTURE struct;
554
```


Figure 5

Figure 5B

Start

610 packages required for signal types?

No

Yes

620 Write library and use clases

630 Write Entity

640 Write Architecture body outline

650 Write signal declarations based on row header entries

B

660 More components ?

No

A

Yes

670 This component already described?

Yes

C

Figure 6A

C

Write component
dec body outline   **675**

D →

680

Yes

More
ports?   B

685

Specified
Port
name?

687   Write port dec.
with synapse
name

Write Port Dec
with signal header
name   **689**

D   D

A

Write
Component
instantiations   **692**

Done

Figure 6B

720   **705**   **702**

	state_machine	F {A:B}	controller
inSig1	I		I
inSig2	I		I
run[6:5]	O	win[6:5]	
config[1:0]	O	end[1:0]	
outSig{A:B}		z	O

725
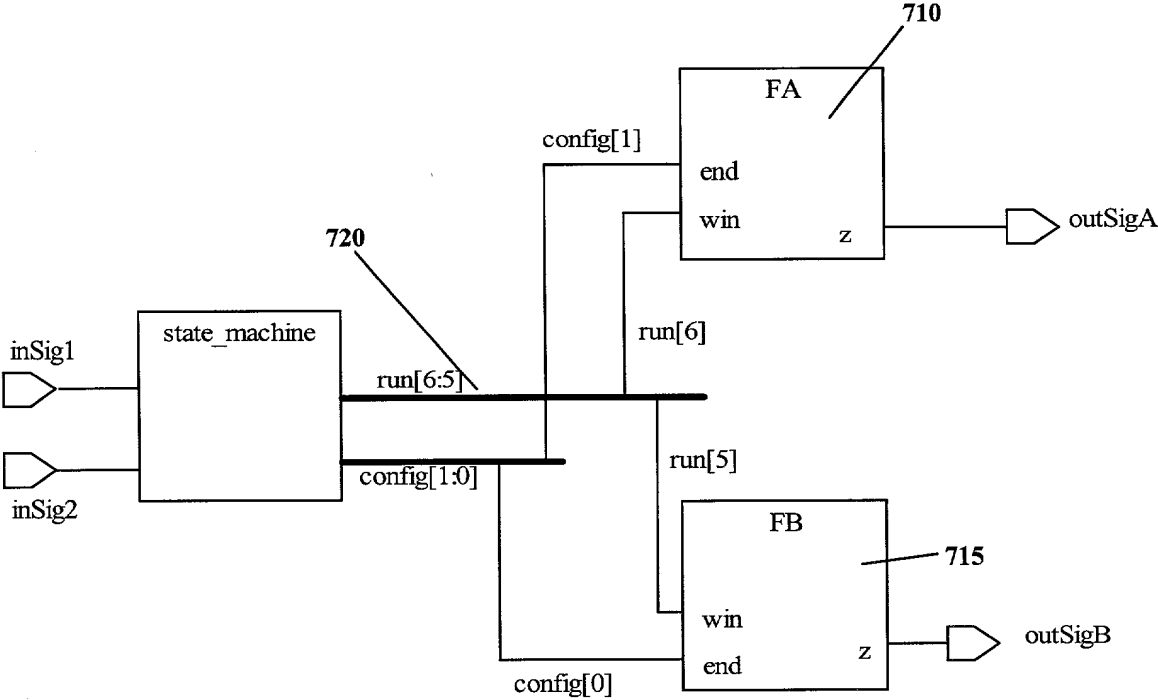
Figure 7A

Figure 7B

Signal	Module	Synapse	Function
Signal	**Module**	I, O, B	*Signal* is connected to module **Module** and the port name is *Signal*
		I, O, B [synapseRange]	**ILLEGAL**
		I, O, B {synapseRange}	**ILLEGAL**
	Module {moduleRange}	I, O, B	*Signal* is connected to all arrayed modules **Modules** and the port name is *Signal*
		I, O, B [synapseRange]	**ILLEGAL**
		I, O, B {synapseRange}	**ILLEGAL**

Figure 8

Signal	Module	Synapse	Function
Signal [range]	**Module**	I, O, B	*Signal* [range] is connected to module **Module** and the port name is *Signal* [range]
		I, O, B [synapseRange]	[synapseRange] must be a member of *signal* [range]. If this criterion is met, *signal* [synapseRange] is connected to **Module** and the port name is *signal* [synapseRange]. See note 3.
		I, O, B {synapseRange}	**ILLEGAL**
	Module {moduleRange}	I, O, B	*Signal* [range] is connected to all arrayed modules **Module** and the port name is *Signal* [range]
		I, O, B [synapseRange]	[synapseRange] must be a member of *signal* [range]. If this criterion is met, *signal* [synapseRange] is connected to all arrayed modules **Module** and the port name is *signal* [synapseRange]. See Note 3.
		I, O, B {synapseRange}	{synapseRange} must be a member of *signal* [range] and its range must be equal to {moduleRange}. If this criterion is met, each arrayed module **Module** has a port named *signal* and they are connected as per Note 1.

Note 1 — The arrayed connectivity works as if both array ranges were expanded, and an order of appearance association is made. For example, if range #1 is A:H and range #2 is 7:0, the ranges would be expand as A, B, C, D, E, F, G, H and 7, 6, 5, 4, 3, 2, 1, 0 respectively. The ranges would be interconnected such that element H in range #1 is connected to element 0 in range #2. This rule is true regardless of whether the range is specified numerically or alphabetically.

Note 2 — An example of this expansion is *mySignalData*{A:D} [7:0]. This signal would be expanded to be *mySignalData*A[7:0], *mySignalData*B[7:0], *mySignalData*C[7:0], and *mySignalData*D[7:0].

Note 3 — If [synapseRange] is only one element, the port name reduces to *signal*.

Figure 9

Signal	Module	Synapse	Function
Signal {array}	**Module**	I, O, B, U	*Signal* {array} is connected to module **Module** with a port for each member in {array}. The ports are individual signals whose names are formed by expanding {array} as per Note 2.
		I, O, B, U [synapseRange]	**ILLEGAL**
		I, O, B, U {synapseRange}	{synapseRange} must be a member of {array}. If this criterion is met, *signal* {synapseRange} is connected to module **Module** with a port for each member in {synapseRange}. The ports are individual signals whose names are formed by expanding {synapseRange} as per Note 2.
	Module {moduleRange}	I, O, B, U	{array} must be equal in range to {moduleRange}. If this criterion is met, each arrayed module **Module** has a port name *signal* and they are connected as per Note 1.
		I, O, B, U [synapseRange]	**ILLEGAL**
		I, O, B, U {synapseRange}	{synapseRange} must be a member of {array} and its range must be equal to {moduleRange}. If this criterion is met, each arrayed module **Module** has a port named *signal* and they are connected as per Note 1.

Note 1 — The arrayed connectivity works as if both array ranges were expanded, and an order of appearance association is made. For example, if range #1 is A:H and range #2 is 7:0, the ranges would be expand as A, B, C, D, E, F, G, H and 7, 6, 5, 4, 3, 2, 1, 0 respectively. The ranges would be interconnected such that element H in range #1 is connected to element 0 in range #2. This rule is true regardless of whether the range is specified numerically or alphabetically.

Note 2 — An example of this expansion is *mySignal*{A:D}*Data*[7:0]. This signal would be expanded to be *mySignalAData*[7:0], *mySignalBData*[7:0], *mySignalCData*[7:0], and *mySig-nalDData*[7:0].

Figure 10

Signal	Module	Synapse	Function
Signal {array} [range]	**Module**	I, O, B	*Signal* {array} [range] is connected to module **Module** with a port for each member in {array}. The ports are [range] vectors whose names are formed by expanding {array} as described in Note 2.
		I, O, B [synapseRange]	[synapseRange] must be a member of [range]. *Signal* {array} [synapseRange] is connected to module **Module** with a port for each member in {array}. The port names are formed by expanding {array} as per Note 2.
		I, O, B {synapseRange}	{synapseRange} must be a member of {array}. If this criterion is met, *signal* {synapseRange} [range] is connected to module **Module** with a port for each member in {synapseRange}. The ports are [range] vectors whose names are formed by expanding {synapseRange} as per Note 2.
	Module {moduleRange}	I, O, B	{array} must be equal in range to {moduleRange}. If this criterion is met, each arrayed module **Module** has a port name *signal* [range] and they are connected as per Note 1.
		I, O, B [synapseRange]	[synapseRange] must be a member of [range] and {array} must be equal in range to {moduleRange}. If this criterion is met, each arrayed module Module has a port named *Signal* {synapseRange} and they are connected as per Note 1. See also Note 3.
		I, O, B {synapseRange}	{synapseRange} must be a member of {array} and its range must be equal to {moduleRange}. If this criterion is met, each arrayed module **Module** has a port named *signal* [range] and they are connected as per Note 1.

<u>Notes</u>

Note 1 — The arrayed connectivity works as if both array ranges were expanded, and an order of appearance association is made. For example, if range #1 is A:H and range #2 is 7:0, the ranges would be expand as A, B, C, D, E, F, G, H and 7, 6, 5, 4, 3, 2, 1, 0 respectively. The ranges would be interconnected such that element H in range #1 is connected to element 0 in range #2. This rule is true regardless of whether the range is specified numerically or alphabetically.

Note 2 — An example of this expansion is *mySignal*{A:D}*Data*[7:0]. This signal would be expanded to be *mySignalAData*[7:0], *mySignalBData*[7:0], *mySignalCData*[7:0], and *mySig-nalDData*[7:0].

Figure 11

	Top	Module1	Module2	Delay	Comment
in1	I	I		20 ns	Input port for top
intSig1		O	I	10 ns	
intSig2		O	I	5 ns	
out1	O		O	5 ns	Output port for top
cellName		Rev3			

Figure 12

```
LIBRARY ieee;
USE ieee.std_logic.all;

ENTITY Top IS
        PORT (in1    : IN    std_logic; -- Input port for top        1310
               out1   : OUT std_logic); -- Output port for top        1320
        ATTRIBUTE delay OF IN1 : SIGNAL IS 20 ns;                    1330
        ATTRIBUTE delay OF OUT1 : SIGNAL IS 5 ns;                    1340
END Top;

ARCHITECTURE struct OF Top IS

        SIGNAL intSig1       : std_logic;
        SIGNAL intSig2       : std_logic;
        ATTRIBUTE delay OF intSig1 : SIGNAL IS 10 ns;                1350
        ATTRIBUTE delay OF intSig2 : SIGNAL IS 5 ns;                 1360


        COMPONENT Module1
        PORT(
            in1          : IN    std_logic;
            intSig1      : OUT std_logic;
            intSig2      : OUT std_logic;
        );
        END COMPONENT;

        ATTRIBUTE cellName of U0 : LABEL IS "Rev3";                  1370

        COMPONENT Module2
        PORT(
            intSig1      : IN    std_logic;
            intSig2      : IN    std_logic;
            out1         : OUT std_logic;
        );
        END COMPONENT;

BEGIN

    U0 : Module1
        PORT MAP(
            in1     => in1,
            intSig1    => intSig1,
            intSig2    => intSig2
        );
```
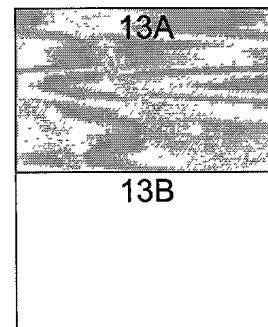

Figure 13

Figure 13A

```
U1 : Module2
    PORT MAP(
        out1    => out1,
        intSig1   => intSig1,
        intSig2   => intSig2
    );

END ARCHITECTURE struct;
```
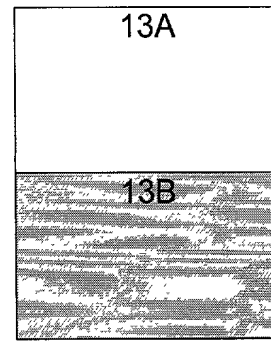

Figure 13

Figure 13B

	Example	Control	Function_one(A:B)	Timing Model	Delay
clock	I	I	I	CK	20
myReq[2:0]		O	I{2:1}	RC	5
config[2:0]	O	O	I	RC	3
ack(A:C)		I	O{A:B}	RC	7
status(A:B)[7:0]		I	O	RC	3
data(0:2)[15:0]		O	I{0:1}	RC	9

◄ ◄ ► ►I \ **Control_IF** / Other_Function_IF / ◄ ► ►I

Figure 14

	Example	Other_Function	Monitor	Timing Model	Delay
clock	I	I	I	CK	20
myReq[2:0]		I[0]	I[2:1]	RC	5
ack(A:C)		O{C}		RC	7
status(A:B)[7:0]		I[3:0]	I{B}	RC	3

◄ ◄ ► ►I \ Control_IF \ **Other_Function_IF** / ◄ ► ►I

Figure 15

	Example	Control	Function_one(A:B)	Other_Function	Monitor	Check[2:0]	Timing Model	Delay
clock	I	I	I	I	I	I	CK	20
myReq[2:0]		O	I{2:1}	I[0]	I[2:1]		RC	5
config[2:0]	O	O	I			I[5:3]	RC	3
ack{A:C}		I	O{A:B}	O{C}		I	RC	7
status{A:B}[7:0]		I	O	I[3:0]	I{B}		RC	3
data{0:2}[15:0]		O	I{0:1}			I[15:8]	RC	9

Figure 16

Figure 17

Common Graphical User Interface (GUI) 1850	Interface based design engine and GUI 1810	Design integrator and netlist writer 1860
	State diagram editor engine and GUI 1840	
	Block diagram editor engine and GUI 1820	
	HDL text editor engine and GUI 1830	

Figure 18

		ModuleName
myBundle	data[32]	B
	R/W*	I

1910

Figure 19

	ModuleName
myBundle	*

Figure 20